CRITERIA FOR FOCUSED DATA COLLECTION

Prepared by
William M. Wilson,
Senior Scientific Specialist

Submitted to: Charles Coleman
Manager, SATC

March, 2001

Technical POC: Dr. Linda Rosenberg
Phone #: 301-286-0087
Fax #: 301-286-1701
Email: Linda.Rosenberg@gsfc.nasa.gov
Mail Code: 302

Administrative POC: Dennis Brennen
Phone #: 301-286-6582
Fax #: 301-286-1667
Email: dbrennan@pop300.gsfc.nasa.gov
Mail Code: 300

CRITERIA FOR FOCUSED DATA COLLECTION

1. Objective

The objective of this task is to propose and validate a mechanism whereby projects can identify their needs for software measurement data and focus their data collection activities using a minimum standardized set of software measures. The purpose of this strategy is to evolve a process that will enable NASA projects to tailor with their data collection activities to their unique needs for effective management control indicators, but also encourages consistent data collection that will facilitate statistical analysis across NASA domains.

2. Background

Both the cost of producing software and the quality characteristics of software products are determined by a variety of project circumstances and events. As yet no common standards have emerged for estimating software development costs, determining the status of software activities, evaluating the effectiveness of software procedures or predicting quality attributes of the software being developed.

The NASA environment emphasizes project autonomy, technical leadership, lean budgets, and close working relationships between Government and contractor organizations. Almost all NASA projects use similar management methodologies and procedures (i.e. formal plans, Work Breakdown Structures (WBS), life cycle reviews, cost accounting, configuration management, nonconformance reporting & corrective action tracking). Many projects also perform statistical defect modeling to track software reliability growth. Most NASA software contractors attempt to improve their development process using fault causal analysis. As a result of these practices there is a large amount of software data generated within NASA's software activities.

However, most NASA projects tend to have unique missions that are accomplished by a diverse mixture of organizations using a variety of technologies. As a result, each project's development environment and its management and control practices are tailored to reflect the organizational mix, the style of the project manager and the availability of resources. This has lead to a profusion of data collection and reporting schemes that are similar in methodology but unique in form, content, and level of detail. In addition, most projects do not retain development data or turn it over to others for postmortem analysis. Consequently, most of NASA's data is not readily available and the data that is available is not consistent in form, content, or level of detail.

3. Problem Definition

The formulation of sound software engineering policy and practices must be based on statistical analysis of data that is accurate and consistent. However, there are no accepted standard methods for defining, measuring and documenting software environments, products and processes. The autonomous nature of NASA projects requires that the introduction of new standards and practices or changes to those currently in use must be demonstratively more effective and come with minimum impact. The problem to be resolved by this task is to establish an effective set of software measures that readily adapted for use by any NASA project.

Software costs, qualities, and delivery schedule are determined by a variety of factors. The most obvious of these factors are;

- the product's required capabilities and attributes,
- the development process' maturity and effectiveness,
- availability and use of appropriate resources and facilities,
- the project staff's experience and training, and
- the nature and structure of project.

Although each of these factors can be quantified, the units of measure and the measurements themselves are often discretionary and biased by the experience of the project staff. As a consequence most project data, although useful within the context of the project, is difficult to correlate with data from other projects.

Due to the unique nature of NASA projects, an acceptable software data collection scheme must fit within existing management methodologies. It must also have minimum impact on project costs and stakeholder prerogatives, in addition to providing effective management insight into ongoing activities. If data is to be collected not only to monitor and control immediate project cost, schedule and product quality, but also to support post-mortem and long-term statistical studies, the burden of collecting such data must be minimal. Therefore, criteria for collecting software data must be primarily focused on those items that can provide useful metrics to management and also be used for long-term statistical analysis.

Therefore, the challenge inherent in this task is to establish a set of measures that will be used by NASA projects to manage software development and at the same time provide data with sufficient scope, detail, and consistency to enable research across various NASA activities.

4. Assumptions

It is assumed that current NASA project methodologies can be augmented and extended to enhance management and control capabilities, enable cross-environment research and still permit tailoring by projects to accommodate unique situations.

5. Approach

5.1 Task Strategy And Document Organization

The strategy embodied by this task is to provide a hierarchy of questions related to project management issues that can be used to focus a project's data collection activities on measures that provide insight into the issues of greatest concern. The hierarchy of issues, key questions and categories of measures is provided below by Section 6 of this document.

Each issue, its related categories of measurement, specific measures within each category, and questions that may be used to focus on specific measures are presented by the subsections of Section 6. Each subsection includes a table of focus questions and corresponding uniquely numbered specific measures. The unique ID number of each specific measure is used find its detailed information in Appendix A.

Appendix A to this document provides a comprehensive set of measures. Each measure is presented as follows:

ID Number Title Of Specific Measure

Brief description of the purpose and nature of the measure.	
Focus Questions	
Typical management and technical questions that the measure may be used to satisfy.	
Data Items Collected:	**Data Source/Criteria**
• Primitive date elements of the measure •	• Typical project products, activities or events that are potential sources of the primitive data elements or are the criteria that validate the data.

Appendix B is a table of specific measures versus questions – i.e. the inverse of the tables contained in Section 6.

Appendix C to this document provides a set of data collection forms that support Appendix A's measures.

5.2 Procedure

The following steps will be taken to expand and refine the set of metrics and measures recommended for NASA projects:

1. A NASA project will be identified as a pilot activity.
2. The objectives and characteristics of the project and the characteristics of the project's development process will be documented using the reporting forms provided by Appendix A.
3. Based on the project's priority of issue(s), the key questions provided below by Table 6.0 will be used to identify a minimum set of measurement categories that support the project's major concern(s).

4. Within each selected measurement category, focus questions will be used to identify the specific measures necessary to provide insight into the software development process on the issues of primary concern.
5. Having selected the specific measures, the following will be determined:
    - What data is required?
    - When is it required?
    - Who collects it?
    - How is it collected?
    - Where and how is it kept?
    - When is it reported?
    - Who has access to the data?
    - How will the measurements be used?
6. The SATC will maintain continuous liaison with the project throughout the software development process to assist in resolving data collection and analysis issues.
7. Based on the pilot project experience, the measures recommended for collection by NASA projects will be enhanced and refined.
8. The data collected by the pilot project and the profile of the project and its development process will be used as an initial entry in the SATC's data research database.

5.3 Rationale & Considerations

Recent engineering research studies have shown that a simple classification scheme using unique categories to characterize software defects can provide insight into development activities and provide in-process feedback. When defects are reported using these categories it is possible to relate changes in the shape of the reliability growth curve to defects of a specific type. Since defect types are associated with specific development activities, defects of a specific type are due to some cause in the process. This enables the shape of the reliability growth curve to suggest development areas that need improvement. For example, a large number of functional defects imply that the design process is faulty.

The success of these studies was due to a standardized vocabulary for software defect reporting and unique, non-overlapping defect categories that encompass the entire development process. Recognizing the importance of this example, particular care will be exercised when assisting the pilot project to define data items to be collected.

Management's fundamental concerns are producing products of the required quality on time and within budget. Software measurement is an effective means to providing the information necessary to identify and manage issues inherent in every project. Consequently, collecting data that provides visibility for product status, resource expenditures, and scheduled accomplishments are readily supported by management. The data that is assembled as part of the Problem/Nonconformance Reporting and Corrective Action (P/NRCA) and Configuration Management (CM) systems are precisely the same data that characterizes the attribute values of software being produced. Although these processes facilitate the data collection aspects of measuring software problems, defects and changes, the variety of finding activities and types of reports make it difficult to communicate clearly and precisely. During software development the assurance activities, whose function is to preclude and find problems, are the primary originators of Nonconformance reports. Planning, designing, technical documentation, and coding activities are also sources of problems reports. Technical staff performing these activities will often generate a Nonconformance report when they discover what appears to be a defect in a software artifact they are using to perform their task. Changes come about as a consequence of resolving a reported nonconformance or in response to a change in mission requirements. Since each technical discipline has a different background and view of the development process, consistent and accurate reporting of nonconformance requires that all parties to the process have a common understanding of the elements of that process. Due to the close coupling of the problem reporting and corrective action and change management systems, it is also necessary the common elements of these methodologies have unique and unambiguous definitions.

Due to the ingrained nature of current practices within NASA projects, it is essential that the pilot project's current practices be capitalized upon to the maximum extent and project personnel be given adequate training and information on any significant departures from existing practices or definitions of terms.

Research's fundamental concerns, in contrast to management's interests, are identifying cause and effect relationships between elements of development environments, software practices and product quality attributes. Although research interests would appear to require capturing data that project management would consider extraneous, most if not all of that data is developed during project planning and implementation. *The information that characterizes the pilot project and its development environment will have to be transposed from the project's plans into the appropriate data collection forms by the SATC.*

6. Collection Rational & Recommended Data Elements

The fundamental reason for measuring software and the software process is to obtain data that will help project management address and resolve resource and product issues. Software measurement also provides a basis for effective communications within the project and justifying project decisions to higher management levels

There are six fundamental issues that are common to all development activities. They are:

1. Schedule and Progress
2. Resources and Cost
3. Growth and Stability
4. Product Quality
5. Development Performance
6. Technical Adequacy

Successful, effective management requires continuous visibility of each of these issues so that timely and informed adjustments can be made to schedules, budgets, and processes. Frequent sampling and assessment of project measurement data provides that visibility.

For each of the fundamental issues there are key questions that the project manager must periodically ask to ensure that the project remains on course and under control. To answer these questions, specific categories of measurement data must be available to the project manager. The issues, key questions related to each issue, and categories of measures necessary to answer the questions are show in Table 6.0.

Issue	Key Questions	Measurement Category
1. Schedule & Progress	Is the project meeting scheduled milestones?	1.1 Milestone Performance
	How are specific activities and products progressing?	1.2 Work Unit Progress
	Is project spending meeting schedule goals?	1.3 Schedule Performance
	Is capability being delivered as scheduled?	1.4 Incremental Capability
2. Resources & Cost	Is effort being expended according to plan?	2.1 Effort Profile
	Are qualified staff assigned according to plan?	2.2 Staff Profile
	Is project spending meeting budget objectives?	2.3 Cost Performance
	Are necessary facilities and equipment available as planned?	2.4 Environment Availability
3. Growth & Stability	Are the product size and content changing?	3.1 Product Size & Stability
	Are the functionality and requirements changing?	3.2 Functional Size & Stability
	Is the target computer system adequate?	3.3 Target Computer Resource Utilization
4. Product Quality	Is the software good enough for delivery?	4.1 Defect Profile
	Is the software testable and maintainable?	4.2 Complexity
5. Development Performance	Will the developer be able to meet budget and schedules?	5.1 Process Maturity
	Is the developer efficient enough to meet current commitments?	5.2 Productivity
	How much breakage to changes and errors has to be handled?	5.3 Rework
6. Technical Adequacy	Is the planned impact of the leveraged technology being realized?	6.1 Technology Impacts

Table 6.0

By focusing data collection activities on measurement categories that answer the key issue questions the project can minimize resources devoted to the measurement process.

With the exception of 6.1 Technical Impacts, the following sections of this document identify specific measures within each measurement category. Questions are also listed for each measurement category which can be used to focus project activities on specific measures.

Technical Impact measures (6.1) are unique to project and technology used and therefore do not lend themselves to specification outside the context of the project.

Appendix A to this document provides a tabular presentation for each specific measure. Each presentation provides a brief rational for the use of the specific measure, the focus questions that the measure can be used to answer, and the specific data elements that must be collected in order to use the measure.

Appendix B to this documents provides a table of Specific Measures versus the questions that the measures may be used to answer.

6.1 Schedule & Progress

Most technical developments are schedule driven. Therefore in almost all instances, progress against an established schedule is the first concern of project and higher levels of management. A project that falls behind has only four alternatives for getting back on schedule; revise the schedule, eliminate requirements, lower product quality standards, or add additional resources. Since these alternatives become increasingly more difficult the further behind the project becomes, it is essential that status data be collected and assessed regularly throughout the life of the project. Table 6.1a below, lists the specific measures that can be collected for each category of schedule and progress measurement.

1. Measurement Category	**Specific Measures**
1.1 Milestone Performance	1.1.1 Milestone Dates
1.2 Work Unit Progress	1.2.1 Requirements Allocated 1.2.2 Components Designed 1.2.3 Components Implemented 1.2.4 Components Integrated & Tested 1.2.5 Test Cases Completed 1.2.6 Paths Tested 1.2.7 Requirements Tested 1.2.8 Changes Implemented 1.2.9 Problem Reports Resolved 1.2.10 Audits & Reviews Completed
1.3 Schedule Performance	1.3.1 Schedule Variance
1.4 Incremental Capability	1.4.1 Build Component Content 1.4.2 Build Function Content

Table 6.1a

1 SCHEDULE & PROGRESS	
FOCUS QUESTION	**SPECIFIC MEASURE**
Are component designs being completed on time?	**1.2.2 Components Designed**
Are components being completed on time?	**1.2.3 Components Implemented**
Are components being incorporated as scheduled?	**1.4.1 Build Component Content**
Are components passing their reviews?	**1.2.9 Reviews Completed**
Are costs conforming to projections?	**1.3.1 Schedule Variance**
Are problem reports being closed at an adequate rate?	**1.2.8 Problem Reports Resolved**
Are requirements being tested as scheduled?	**1.2.7 Requirements Tested**
Are reviews being held on schedule?	**1.2.9 Reviews Completed**
Are tests being completed on schedule?	**1.2.5 Test Cases Completed**
Has the requirements test matrix been completed?	**1.2.1 Requirements Allocated**
Have all of the paths been successfully tested?	**1.2.6 Paths Tested**
Have all requirements been allocated to at least one design component?	**1.2.1 Requirements Allocated**
Have the tests been successful?	**1.2.7 Requirements Tested**
How frequently has the schedule changed?	**1.1.1 Milestone Dates**
How likely is there to be a cost overrun?	**1.3.1 Schedule Variance**
How many activities are scheduled concurrently?	**1.1.1 Milestone Dates**
How many change requests have impacted the software, the schedule, or the budget?	**1.2.10 Changes Implemented**
How many requirements can be directly tested?	**1.2.1 Requirements Allocated**
How many test cases are required to completely test the software?	**1.2.6 Paths Tested**
Is functionality being incorporated as scheduled?	**1.4.2 Build Function Content**
Is integration and testing being accomplished on schedule?	**1.2.4 Components Integrated and Tested**
Is the planned implementation rate realistic?	**1.2.3 Components Implemented**
Is the planned rate of integration and testing realistic?	**1.2.4 Components Integrated and Tested**
Is the production schedule being met?	**1.3.1 Schedule Variance**
Is the rate of change requests decreasing?	**1.2.10 Changes Implemented**
Is the rate of problem reporting going down?	**1.2.8 Problem Reports Resolved**
Is the schedule for component designs realistic?	**1.2.2 Components Designed**
Is the schedule realistic?	**1.1.1 Milestone Dates**
Is the test schedule realistic?	**1.2.5 Test Cases Completed**
What components are behind schedule?	**1.2.3 Components Implemented**
What components have been added, deferred or eliminated?	**1.4.1 Build Component Content**
What components have failed their review?	**1.2.9 Reviews Completed**
What functional tests are behind schedule?	**1.2.7 Requirements Tested**
What functionality is being deferred?	**1.4.2 Build Function Content**
What functions have not been tested?	**1.2.5 Test Cases Completed**
What is the risk of not delivering on schedule?	**1.1.1 Milestone Dates**
What percent of the functionality has been tested?	**1.2.7 Requirements Tested**
What percentage of the paths have been tested?	**1.2.6 Paths Tested**
When will testing be complete?	**1.2.8 Problem Reports Resolved**
Which components have the most open problem reports?	**1.2.8 Problem Reports Resolved**
Will each increment contain the specified components?	**1.4.1 Build Component Content**
Will each increment contain the specified functionality?	**1.4.2 Build Function Content**

Table 6.1b

6.2 Resources & Cost

Tracking and projecting the expenditure and availability of resources against planned and current allocations is the second highest priority for most project managers. The four categories of measures and eight specific measures listed by Table 6.2a can be used to assess expenditure rates and allocation effectiveness..

Measurement Category	Specific Measures
2.1 Effort Profile	2.1.1 Effort
2.2 Staff Profile	2.2.1 Staff Allocation 2.2.2 Staff Experience 2.2.3 Staff Turnover
2.3 Cost Performance	2.3.1 Cost Variance 2.3.2 Cost Profile
2.4 Environment Availability	2.4.1 Resource Availability Dates 2.4.2 Resource Utilization

Table 6.2a

2 RESOURCES & COST	
FOCUS QUESTION	**SPECIFIC MEASURE**
Are certain activities or functions requiring more staff than expected?	**2.2.1 Staff Allocation**
Are development resources being applied according to plan?	**2.1.1 Effort**
Are key resources to be available when needed?	**2.4.1 Resource Availability Dates**
Are program costs in accordance with budgets?	**2.3.2 Cost Profile**
Are project costs in accordance with the budget?	**2.3.1 Cost Variance**
Are specific tasks or activities taking more/less effort than expected?	**2.1.1 Effort**
Are sufficient development resources available and allocated for each activity?	**2.2.1 Staff Allocation**
Are sufficient experienced/trained personnel available?	**2.2.2 Staff Experience**
Are the available resources sufficient?	**2.4.2 Resource Utilization**
How are experience levels being affected by turnover?	**2.2.3 Staff Turnover**
How effectively are resources being used?	**2.4.2 Resource Utilization**
How many people have been added or left the project?	**2.2.3 Staff Turnover**
Is the availability of resources impacting progress?	**2.4.1 Resource Availability Dates**
Is the effort profile realistic?	**2.1.1 Effort**
What areas are most affected by turnover?	**2.2.3 Staff Turnover**
What is the projected completion cost?	**2.3.1 Cost Variance**
What WBS elements or tasks have the greatest variance?	**2.3.1 Cost Variance**
Will additional training be required?	**2.2.2 Staff Experience**
Will the project budget be adequate or will there be an overrun?	**2.3.2 Cost Profile**

Table 6.2b

6.3 Growth & Stability

The measurement categories and specific measure listed in Table 6.3a can be used to monitor the size, rage of growth and frequency of change to the various instantiations of the software product throughout the development process. Tracking the stability and growth of requirements is essential to being able to stay on schedule and meet budget plans. Staying abreast of resource utilization and demands is equally important to maintaining control of the product.

Measurement Category	Specific Measures
3.1 Functional Size & Stability	3.1.1 Requirements 3.1.2 Function Points
3.2 Product Size & Stability	3.2.1 Lines of Code 3.2.2 Number of Components 3.2.3 Words of Memory 3.2.4 Database Size
3.3 Target Computer Resource Utilization	3.3.1 CPU Utilization 3.3.2 CPU Throughput 3.3.3 I/O Utilization 3.3.4 I/O Throughput 3.3.5 Memory Utilization 3.3.6 Storage Utilization 3.3 7 Response Time

Table 6.3a

3 GROWTH & STABILITY	
FOCUS QUESTION	**SPECIFIC MEASURE**
Are requirements being deferred to later builds?	**3.1.1 Requirements**
Can additional data traffic be provided after system delivery?	**3.3.3 I/O Utilization**
Can the CPU resources support additional functionality?	**3.3.1 CPU Utilization**
Can the software design handle the required amount of system data in the allocated time?	**3.3.4 I/O Throughput**
Can the software handle additional system data a after delivery?	**3.3.4 I/O Throughput**
Can the software size increase after system delivery as needed to incorporate new functionality?	**3.3.5 Memory Utilization**
Do CPU estimates appear reasonable?	**3.3.1 CPU Utilization**
Do storage estimates appear adequate?	**3.3.6 Storage Utilization**
Do the I/O resources allow adequate data traffic flow?	**3.3.3 I/O Utilization**
Does the memory need to be upgraded?	**3.2.3 Words of Memory**
Does the software operate efficiently?	**3.3.7 Response Time**
Has the size allocated to each build changed?	**3.2.1…Lines of Code**
Have components allocated to each build changed?	**3.2.2 Number of Components**
Have large increases in CPU utilization occurred?	**3.3.2 CPU Throughput**
Have requirements allocated to each incremental build changed?	**3.1.1 Requirements**
Have sufficient CPU resources been Acquired?	**3.3.2 CPU Throughput**
Have sufficient CPU resources been provided?	**3.3.1 CPU Utilization**
Have sufficient storage resources been provided?	**3.3.6 Storage Utilization**
How accurate was the size estimate that schedule and effort plans were based on?	**3.2.1 Lines of Code**
How long do certain services take?	**3.3.7 Response Time**
How many components need to be implemented and tested?	**3.2.2 Number of Components**
How many different data types have to be addressed?	**3.2.4 Database Size**
How much data has to be handled by the system?	**3.2.4 Database Size**
How much functionality is in the software?	**3.1.2 Function Points**
How much has the approved software baseline changed?	**3.2.2 Number of Components**
How much has the software changed?	**3.2.1 Lines of Code**
How much spare memory capacity is available?	**3.2.3 Words of Memory**
How much work is to be done?	**3.1.2 Function Points**
How mush has software functionality changed?	**3.1.1 Requirements**
Is functionality slipping to later builds?	**3.2.2 Number of Components**
Is the target computer system sufficient to meet response requirements?	**3.3.7 Response Time**
Should I/O resources be expanded?	**3.3.3 I/O Utilization**
What components are affected by the changes?	**3.1.1 Requirements**
What components have grown or gotten smaller?	**3.2.1 Lines of Code**
What is the expansion capacity?	**3.3.6 Storage Utilization**
What is the risk that system errors will be caused by lack of storage space?	**3.3.5 Memory Utilization**
Will the software fit in the processors?	**3.3.5 Memory Utilization**

Table 6.3b

6.4 Issue: Product Quality

Quality must be designed and built into the software product during development. It cannot be tested into the product or added on after development. If product quality is a issue of primary concern, it must be monitored throughout the development lifecycle. The measurement categories and specific measures listed by Table 6.4a can be used to provide the visibility needed to monitor product quality.

Measurement Category	Specific Measures
4.1 Defect Profile	4.1.1 Problem Report Trends 4.1.2 Problem Report Aging 4.1.3 Defect Density 4.1.4 Failure Interval
4.2 Complexity	4.2.1 Cyclomatic Complexity 4.2.2 Weighted Methods Per Class 4.2.3 Response For A Class 4.2.4 Lack of Cohesion 4.2.5 Coupling Between Object Classes 4.2.6 Depth Of Inheritance 4.2.7 Number Of Children

Table 6.4a

4 PRODUCT QUALITY	
Are difficult problems being deferred?	**4.1.2 Problem Report Aging**
Are reported problems being closed in a timely manner?	**4.1.2 Problem Report Aging**
Do report arrival and closure rates support the scheduled completion date of integration and test?	**4.1.1 Problem Report Trends**
FOCUS QUESTION	**SPECIFIC MEASURE**
How complex are the methods within each class of objects?	**4.2.4 Lack of Cohesion of Methods**
How long does it take to close a problem report?	**4.1.2 Problem Report Aging**
How many complex classes of objects are in this program?	**4.2.2 Weighted Methods Per Class**
How many complex components exist in this program?	**4.2.1 Cyclomatic Complexity**
How many methods are contained within each class?	**4.2.3 Response For A Class**
How many objects should be subjected to additional testing	**4.2.2 Weighted Methods Per Class**
How many problem reports are open? What are their priorities?	**4.1.1 Problem Report Trends**
How many problems reports have been written?	**4.1.1 Problem Report Trends**
How much code is being reused?	**4.2.6 Depth Of Inheritance**
How much code is being reused?	**4.2.7 Number Of Children**
How often will software failures occur during operation of the system?	**4.1.4 Failure Interval**
How reliable is the software?	**4.1.4 Failure Interval**
What are the most complex components?	**4.2.1 Cyclomatic Complexity**
What classes are the most complex?	**4.2.2 Weighted Methods Per Class**
What components are candidates for rework?	**4.1.3 Defect Density**
What components have a disproportionate amount of defects?	**4.1.3 Defect Density**
What components require additional testing or review?	**4.1.3 Defect Density**
What components should be subject to additional testing?	**4.2.1 Cyclomatic Complexity**
What is the program's expected operational reliability?	**4.1.4 Failure Interval**
What is the quality of the software?	**4.1.3 Defect Density**
What objects are most complex?	**4.2.2 Weighted Methods Per Class**
Which object classes are tightly coupled?	**4.2.5 Coupling Between Object Classes**
Which object classes will be difficult to reuse?	**4.2.5 Coupling Between Object Classes**
Which object classes will be most difficult to maintain?	**4.2.5 Coupling Between Object Classes**
Which object classes will be most difficult to maintain?	**4.2.6 Depth Of Inheritance**
Which object classes will be most difficult to test?	**4.2.6 Depth Of Inheritance**
Which object classes' methods will require the most testing?	**4.2.7 Number Of Children**
Which objects have the most externally invoked methods?	**4.2.3 Response For A Class**
Which objects should be subjected to additional testing?	**4.2.3 Response For A Class**
Which variables are referenced by how many methods?	**4.2.4 Lack of Cohesion of Methods**
Within each class, how many methods reference each variable?	**4.2.4 Lack of Cohesion of Methods**

Table 6.4b

6.5 Issue: Development Performance

The primary focus of a software engineering organization is to produce a product that satisfies its requirements on time and within budget. A major goal of management is to improve the engineering organization's effectiveness. Improvement requires definition of the current processes, calibration of past performance and monitoring current rates of productivity. The measurement categories and specific measures listed in Table 6.5a may be used to provide the information necessary to assess a processes need to be improved.

Measurement Category	**Specific Measures**
5.1 Process Maturity	5.1.1 Capability Maturity Model Level
5.2 Productivity	5.2.1 Product Size/Effort Ratio 5.3.2 Functional Size/Effort Ratio
5.3 Rework	5.3.1 Rework Size 3.3.2 Rework Effort

Table 6.5a

5 DEVELOPMENT PERFORMANCE	
FOCUS QUESTION	**SPECIFIC MEASURE**
How efficiently is software being produced?	**5.2.1 Product Size/Effort Ratio**
How efficiently is software being produced?	**5.2.2 Functional Size/Effort Ratio**
How much code had to be changed as a result of correcting defects?	**5.3.1 Rework Size**
How much effort was expended on fixing defects in the software product?	**5.3.2 Rework Effort**
Is product being developed at a rate to be completed within budget?	**5.2.1 Product Size/Effort Ratio**
Is product being developed at a rate to be completed within budget?	**5.2.2 Functional Size/Effort Ratio**
Is the amount of rework impacting cost or schedule?	**5.3.2 Rework Effort**
Is the amount of rework impacting the cost and schedule?	**5.3.1 Rework Size**
Is the developer's software process adequate to address anticipated program risks, issues, and constraints?	**5.1.1 Capability Maturity Model Level**
Is the planned software productivity rate realistic?	**5.2.1 Product Size/Effort Ratio**
Is the planned software productivity rate realistic?	**5.2.2 Functional Size/Effort Ratio**
What is the developer's current software development capability rating?	**5.1.1 Capability Maturity Model Level**
What project management and software engineering practices can be improved?	**5.1.1 Capability Maturity Model Level**
What software development activity required the most rework?	**5.3.2 Rework Effort**
What was the quality of the initial development effort?	**5.3.1 Rework Size**

Table 6.5b

Appendix A

Appendix B

Appendix C